Generatori leksičkih analizatora

jflex

Leksički analizator

- Razbija ulazni niz karaktera na simbole (tokene).
- Simboli se identifikuju pomoću pravila - regularnih izraza.
- Niz karaktera koji odgovara nekom regularnom izrazu može da se predstavi kao simbol.
- Ako niz karaktera ne odgovara ni jednom regularnom izrazu prijavljuje se greška.

Primer

- Ulaz (niz karaktera)
  - 12abc.xx72
- Izlaz (niz simbola)
  - $NUMBER_{12}$ $WORD_{abc}$ $DOT_{.}$ $WORD_{xx}$ $NUMBER_{72}$

Kako analizator radi?

```
switch (ulaz) {
   case [1-9]+          : return NUMBER;
   case [a-z|A-Z]+      : return WORD;
   case “.”             : return DOT;
   default              : return INVALID;
}
```

Smisao upotrebe

- Pojednostavljuje rad parsera
- Parser proverava generičke konstrukcije:
  - NUMBER PLUS NUMBER MUL NUMBER
    - Umesto 12 + 32 * 7
  - NUMBER NUMBER PLUS
    - 72 61 +
- Jednostavnija sintaksna analiza

Generatori leksičkih analizatora

- Zadaju im se definicije regularnih izraza koje treba prepoznati.
- Zadaju im se definicije vrednosti koje treba vratiti u slučaju da ulaz odgovara nekom regularnom izrazu.
- Generišu izvorni kod klase koja vrši leksičku analizu po specifikaciji koju smo zadali.

jflex

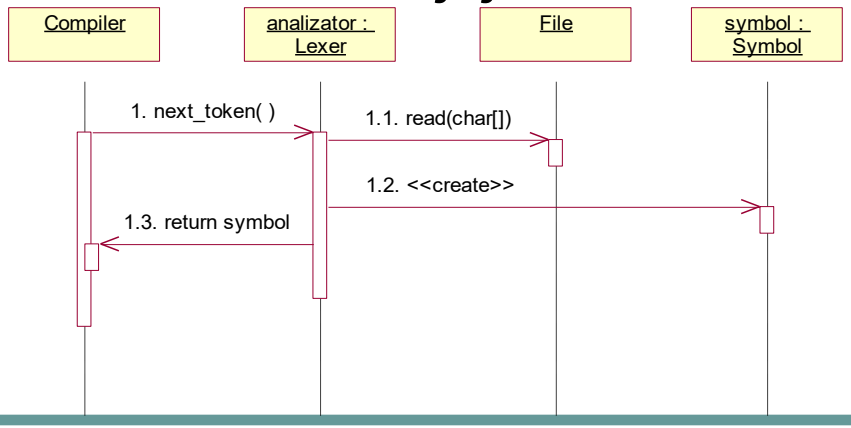
- Generiše izvorni kod leksičkog analizatora u programskom jeziku Java.
- http://jflex.de/
- Pravljen po uzoru na jedan od prvih i najpoznatiji generator leksičkih analizatora – Lex (1975).

Generisanje leksičkog analizatora (jflex)

- Na osnovu tekstualne specifikacije (.flex fajl) generiše se .java fajl koji predstavlja analizator.
- jflex.Main klasa generiše analizator.

Kako se generisani analizator koristi?

- Kompajler traži sledeći simbol (token) od analizatora.
- Analizator čita niz karaktera iz izvora.
- Analizator uparuje niz sa nekim od regularnih izraza.
- Analizator kreira simbol (token) koji odgovara pročitanom ulazu.
- Analizator upisuje u kreirani simbol informacije o ulazu.
- Analizator vraća simbol koji je kreirao.



Koraci u pravljenju analizatora pomoću jflex-a (opšti slučaj)

1. U nekom tekst editoru (npr. WordPad) se napiše specifikacija analizatora koji nam je potreban, pa se sačuva kao .flex fajl (npr. Mjflexer.flex).
2. Napiše se Java klasa (ime te klase po default-u je Yytoken) čije instance će predstavljati tokene.
3. Napiše se Java klasa (klasa sym) koja će enkapsulirati integer konstante koje predstavljaju kodove pojedinačnih tokena (npr. 1 odgovara tokenu za +, 2 tokenu za -, itd.)
4. Pod pretpostavkom da je classpath ispravno postavljen i da radimo u direktorijumu C:\Java, izvorni kod leksičkog analizator ćemo dobiti sledećom komandom:

   C:\Java>java –jar JFlex.jar Mjflexer.flex

   Ovako u direktorijumu C:\Java dobijamo fajl Yylex.java.
5. Kompajliramo redom fajlove sym.java, Yytoken.java i Yylex.java. Dobijamo sym.class, Yytoken.class i Yylex.class koji predstavlja željeni leksički analizator.

Format .lex fajla

import iskazi

%%

jflex direktive

%%

Pravila za uparivanje

Import iskazi

- Služe da se definiše koje pakete treba uvesti u generisanu klasu analizatora.
- Ovi iskazi se prepisuju na početak fajla sa izvornim kodom analizatora.
- NE proverava se sintaksa.

jflex direktive

- Direktive oblika

  %{

  <code>

  %}

  služe da se Java kod naveden u sekciji <code> direktno prepiše u klasu Yylex leksičkog analizatora.
- Delovi %{ i %} moraju biti na početku linije da bi se direktiva prepoznala.
- Opisana direktiva nam daje mogućnost da deklarišemo dodatne promenljive i metode unutar klase leksičkog analizatora kojeg pravimo.

jflex direktive

- Direktive oblika

  %init{

  <code>

  %init}

  služe da se Java kod naveden u sekciji <code> direktno prepiše u konstruktor klase Yylex leksičkog analizatora.
- Delovi %init{ i %init} moraju biti na početku linije da bi se direktiva prepoznala.
- Voditi računa o izuzecima!

jflex direktive

- Direktive oblika

  %eof{

  <code>

  %eof}

  služe da se Java kod naveden u sekciji <code> kopira u klasu Yylex leksičkog analizatora i izvrši čim se dođe do kraja ulaznog fajla kojeg analizator obrađuje.
- Delovi %eof{ i %eof} moraju biti na početku linije da bi se direktiva prepoznala.
- Voditi računa o izuzecima!

jflex direktive

- U odeljku sa direktivama se mogu navoditi i makro definicije.
- Svaka makro definicija se navodi u zasebnoj liniji i sastoji se od makro imena, znaka jednakosti i definicije:

  <name>=<definition>
- Makro imena moraju biti ispravni identifikatori.
- Sama definicija mora biti ispravan regularni izraz, oblika istog kao u delu sa pravilima za uparivanje.
- Definicija može sadržati druge makro ekspanzije, ali uzajamno rekurzivne konstrukcije, tj. ciklusi, u makro definicijama nisu dozvoljeni.

jflex direktive

- U odeljku sa direktivama se mogu navoditi i deklaracije leksičkih stanja. Stanja nam omogućuju da kontrolišemo kada će se određeni regularni izrazi uparivati.
- Oblik direktive:

  %state state[0][, state[1], state[2], …]
- Jedno stanje je implicitno deklarisano u jflex-u. To je stanje YYINITIAL i u njemu analizator počinje analizu.

jflex direktive

- Da bi se obezbedilo da analizator kojeg pravimo broji karaktere u ulaznom fajlu koji se analizira, potrebno je navesti direktivu:

  %char
- Da bi se obezbedilo da analizator kojeg pravimo broji linije u ulaznom fajlu koji se analizira, potrebno je navesti direktivu:

  %line
- Opis ostalih direktiva pogledati u Uputstvu za jflex.

CUP-kompatibilni leksički analizatori

- Realizuju interfejs: java_cup.runtime.Scanner
- Funkcija za analizu se zove next_token
- Klasa koja opisuje tokene nije standardna Yytoken, već java_cup.runtime.Symbol
- Da bi se sve to obezbedilo koristi se deklaracija %cup

Pravila za uparivanje

- Ova pravila čine treći deo .lex fajla.
- Oblik pravila je:

  [<states>] <expression> {<action>}
- Regularni izraz <expression> definiše format ulaza koji treba upariti.
- <action> je Java kod koji se izvršava kada analizator upari ulaz sa regularnim izrazom.
- <states> je opciona lista stanja. Ako se ona nalazi na početku pravila za uparivanje, to znači da će pravilo biti razmatrano samo ako se analizator nalazi u nekom od stanja navedenih u listi.
- Uparivanje se vrši tako što se traži najduži niz karaktera sa ulaza koji odgovara nekom regularnom izrazu. Ako niz karaktera sa ulaza može da se upari sa više pravila, onda se od njih bira pravilo koje je navedeno prvo u .lex fajlu.

Primer pravila za uparivanje

```
%%
[1-9]+      { return 1; }
[a-z]+      { return 2; }
"="         { return 3; }
"+"         { return 4; }
```

Regularni izrazi

- Regularni izrazi ne smeju da sadrže bele karaktere, jer se oni tumače kao kraj izraza. Ako želimo da beli karakter (osim novog reda) označava sebe, potrebno je da ga stavimo pod znake navoda.
- Metakarakteri sa specijalnim značenjem u jflex regularnim izrazima su:

  ? * + | ( ) ^ $. [ ] { } " \
- Ostali karakteri označavaju sami sebe.

Regularni izrazi

Izraz	Opis
pq	Konkatenacija izraza p i q
p|q	Unija (ili) izraza p i q
p?	0 ili 1 pojavljivanje izraza p ($p^0|p^1$)
p+	Pozitivno zatvaranje ($p^1|p^2|p^3|\ldots$)
p*	Zvezdasto zatvaranje ($p^0|\ p^1|p^2|p^3|\ldots$)
"…"	Kada se stave između znaka navoda, metakarakteri gube svoje značenje. Jedini izuzetak je sekvenca \ " (koja označava jedan karakter ")
.	Bilo koji karakter ili niz karaktera, izuzev novog reda

Regularni izrazi

Izraz	Opis
A	Uparuje karakter A
Package	Uparuje reč package
{name}	Uparuje makro definiciju, gde je name ime makroa.
\udddd	Unicode znak čiji kod odgovara heksadecimalnom broju dddd
\xdd	Ascii znak čiji kod odgovara heksadecimalnom broju dd
\ddd	Ascii znak čiji kod odgovara oktalnom broju ddd
\b	Backspace

Regularni izrazi

Izraz	Opis
\n	Novi red
\t	Tab
\f	\Formfeed
\r	Carriage return
$	Kraj linije. Ako se ovaj znak nalazi na kraju regularnog izraza, onda se taj izraz uparuje samo na kraju linije
(…)	Male zagrade služe za grupisanje unutar regularnih izraza

Regularni izrazi

- Regularni izraz […] označava klasu karaktera, pri čemu se uparuje bilo koji karakter od onih navedenih u zagradama. Ako je prvi karakter posle [karakter ^, onda se cela klasa karaktera negira i regularni izraz uparuje sve karaktere osim onih u zagradama.
- Unutar srednjih zagrada važe druga pravila za metakaraktere, pri čemu sledeći izrazi imaju posebno značenje:

{name}	Makro ekspanzija
a-b	Bilo koji karakter između malih slova a i b, uključujući i njih. Ako se – nalazi odmah na početku ili na samom kraju klase karaktera, onda gubi navedeno značenje.
"..."	Svi metakarakteri unutar znaka navoda gube svoje posebno značenje. Jedini izuzetak je sekvenca \ " (koja označava jedan karakter ").
\	Metakarakter koji se navede posle \ gubi svoje posebno značenje.

Regularni izrazi - Primeri

Izraz	**Značenje**
[a-z]	Bilo koje malo slovo
[^0-9]	Sve izuzev cifara
[0-9a-fA-F]	Bilo koja heksadecimalna cifra
[\-\\]	Crtica – ili kosa crta \
["A-Z"]	Uparuje se A ili – ili Z
\x2b	Karakter +
([a-z]|[A-Z])[a-z|A-Z|0-9|_]*	Standardni identifikator
[+-]	Uparuje + ili -
break	Ključna reč break

Akcije

- Predstavljaju Java kod
- Izvršavaju se kada se ulaz upari sa odgovarajucim regularnim izrazom
- Trebalo bi da obezbede vraćanje odgovarajućeg tokena i/ili prelaze među stanjima
- Java kod se prepisuje u generisani analizator bez provere ispravnosti

Metode i promenljive vidljive u akcijama

Element	Opis
String yytext()	Vraća String koji predstavlja trenutno upareni deo ulaza
int yychar	Pozicija karaktera
int yyline	Broj tekuće linije
yybegin(state)	Prelazak u stanje state
yybegin()	Prelazak u početno stanje YYINITIAL